Distributed Database(DDB)

Zain Arshad

UOL Defence Road

zain.arshad@cs.uol.edu.pk

Lecture 12

Relational Algebra

Database Concepts

- A database is a **structured collection** of data related to some real-life phenomena that we are trying to model.
- As an **example** we use a database that models an **engineering company**.

- For each **employee**, we would like to keep track of the **employee number (ENO), name (ENAME), title in the company (TITLE), salary (SAL), identification number of the project(s) the employee is working on (PNO), responsibility within the project (RESP),** and **duration of the assignment to the project (DUR)** in months.

- Similarly, for each **project** we would like to store the **project number (PNO)**, the **project name (PNAME)**, and the **project budget (BUDGET)**.

Sample Database Instance

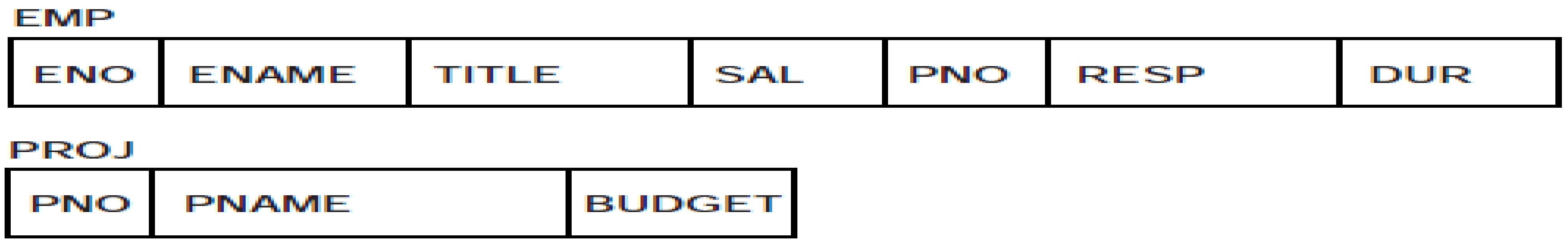
EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Database Terms



EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR

PROJ

PNO	PNAME	BUDGET

Sample Database Scheme

- The entities to be modelled are the **employees** (EMP) and **projects (PROJ)**.
- The relation schemas for this database can be defined as follows:
- **EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR).**
- **PROJ(PNO,PNAME, BUDGET).**
- In relation scheme **EMP**, there are **seven attributes**: **ENO, ENAME, TITLE, SAL,PNO, RESP, DUR.**
- In relation scheme **PROJ**, there are **three attributes**: **PNO, PNAME, BUDGET**

Database Terms

- The number of **attributes** of a relation defines its **degree**, whereas the number of **tuples** of the relation defines its **cardinality**.
- The **empty table**, showing the structure of the table, corresponds to the **relation schema**.

- When the table is **filled with rows**, it corresponds to a **relation instance**.
- Special value of the attribute is generally referred to as the **null value.**
- **Special care** should be given to **differentiate null** from **zero**.

- For example, **value "0"** for **attribute DUR** is **known information** (e.g., in the case of a newly hired employee), while value "**null**" for DUR means **unknown**.

Data Definition Language

- **DDL** is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- **CREATE** - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- **ALTER** - alters the structure of the existing database
- **DROP** - delete objects from the database
- **TRUNCATE** - remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT** - add comments to the data dictionary
- **RENAME** - rename an object

Data Manipulation Language

- DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- **SELECT** - retrieve data from a database
- **INSERT** - insert data into a table
- **UPDATE** - updates existing data within a table
- **DELETE** - Delete all records from a database table
- **MERGE - UPSERT** operation (insert or update)
- **CALL** - call a PL/SQL or Java subprogram
- **EXPLAIN PLAN** - interpretation of the data access path
- **LOCK TABLE** - concurrency Control

- DCL is short name of **Data Control Language** which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

- **GRANT** - allow users access privileges to the database
- **REVOKE** - withdraw users access privileges given by using the GRANT command

- TCL is short name of **Transaction Control Language** which deals with a transaction within a database.

- **COMMIT** - commits a Transaction
- **ROLLBACK** - rollback a transaction in case of any error occurs
- **SAVEPOINT** - to rollback the transaction making points within groups
- **SET TRANSACTION** - specify characteristics of the transaction

Relational Data Languages

- A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database.
- This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases.

- Data manipulation languages developed for the relational model (commonly called query languages) fall into two fundamental groups: relational algebra languages and relational calculus languages. The difference between them is based on how the user query is formulated.

- The **relational algebra** is procedural in that the user is expected to specify, using certain high-level operators, how the result is to be obtained.

- The **relational calculus**, on the other hand, is non-procedural; the user only specifies the relationships that should hold in the result

Relational Algebra

- Relational algebra consists of a set of operators that operate on relations. Each operator takes one or two relations as operands and produces a result relation, which, in turn, may be an operand to another operator.
- These operations permit the querying and updating of a relational database.

- There are **five** fundamental **relational algebra operators** and five others that can be defined in terms of these. The fundamental operators are **selection, projection, union, set difference, and Cartesian product**. The **first two** of these **operators are unary operators**, and the **last three are binary operators**.

- The **additional operators** that can be defined in terms of these fundamental operators **are intersection, θ join, natural join, semi-join and division.**

Relational calculus

- Many operator definitions refer to “formula”, which also appears in relational calculus expressions.
- We define a formula within the context of first-order predicate calculus
- First-order predicate calculus is based on a symbol alphabet that consists of

1. **Variables, constants, functions, and predicate symbols;**
2. **Parentheses;**
3. **Logical connectors ∧(and), ∨(or), ¬(not), →(implication), ↔ (equivalence).**
4. **Quantifiers ∀ (for all) and ∃ (there exists).**

Selection

- Selection produces a horizontal subset of a given relation. The subset consists of all the tuples that satisfy a formula (condition). The selection from a relation R is

  $$\sigma_F(R)$$

  where R is the relation and F is a formula.

  Consider the relation EMP shown in Figure 2.3. The result of selecting those tuples for electrical engineers is

$$\sigma_{\text{TITLE="Elect. Eng."}}(\text{EMP})$$

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E6	L. Chu	Elect. Eng.

Projection

- Projection produces a vertical subset of a relation. The result relation contains only those attributes of the original relation over which projection is performed. Thus the degree of the result is less than or equal to the degree of the original relation.
- The projection of relation R over attributes A and B is denoted as

$$\Pi_{A,B}(R)$$

The projection of relation PROJ shown in Figure 2.3 over attributes PNO and BUDGET

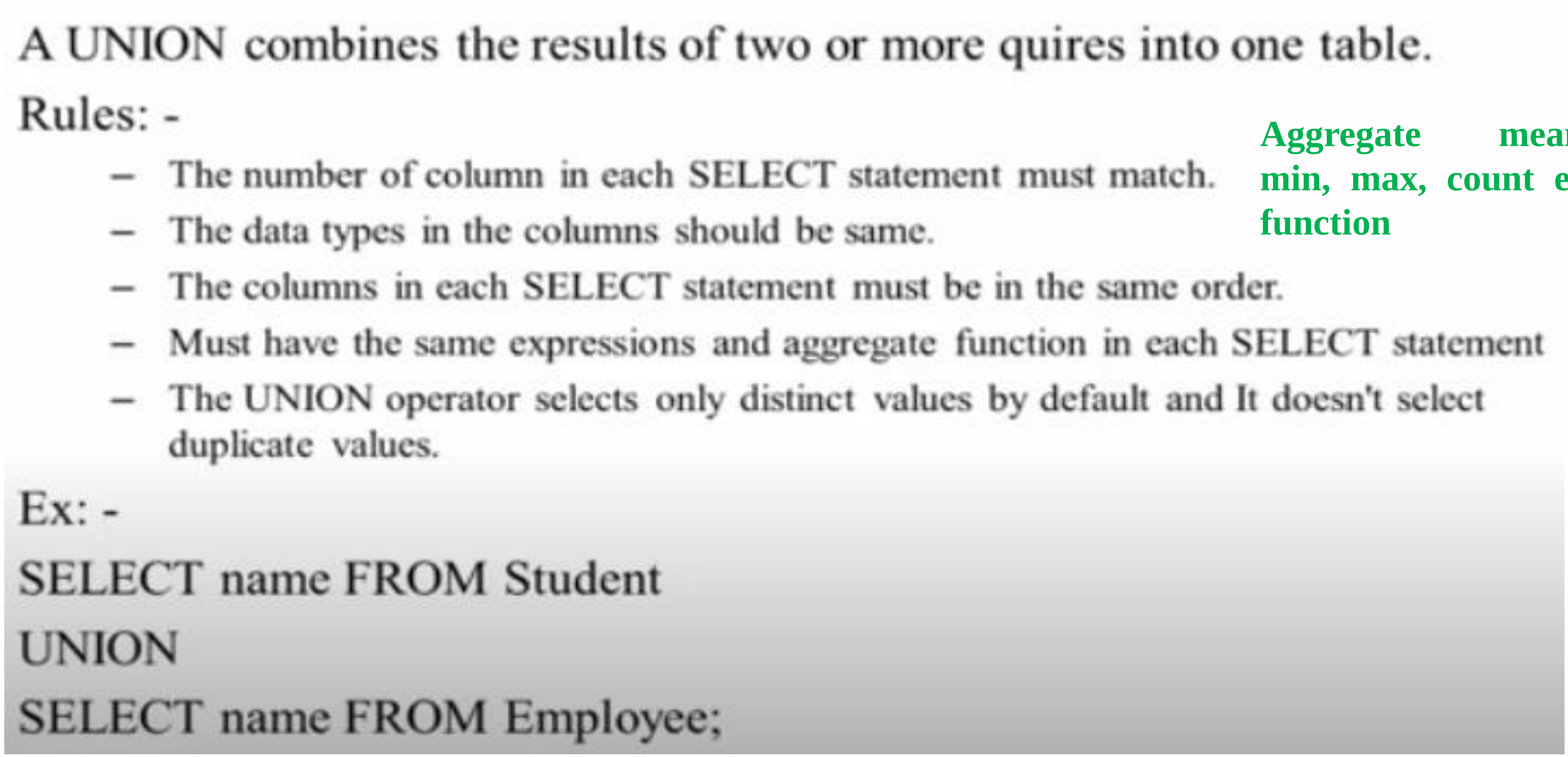
$\Pi_{\text{PNO,BUDGET}}$ (PROJ)

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

Union

- The union of two relations R and S (denoted as R U S) is the set of all tuples that are in R, or in S, or in both.
- We should note that R and S should be union compatible. As in the case of projection, the duplicate tuples are normally eliminated.
- Union may be used to insert new tuples into an existing relation, where these tuples form one of the operand relations.

UNION



A UNION combines the results of two or more quires into one table.

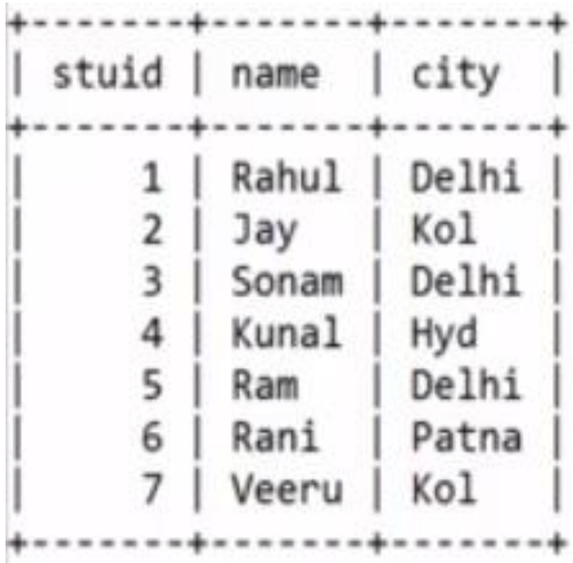
Rules: -

- The number of column in each SELECT statement must match.
- The data types in the columns should be same.
- The columns in each SELECT statement must be in the same order.
- Must have the same expressions and aggregate function in each SELECT statement
- The UNION operator selects only distinct values by default and It doesn't select duplicate values.

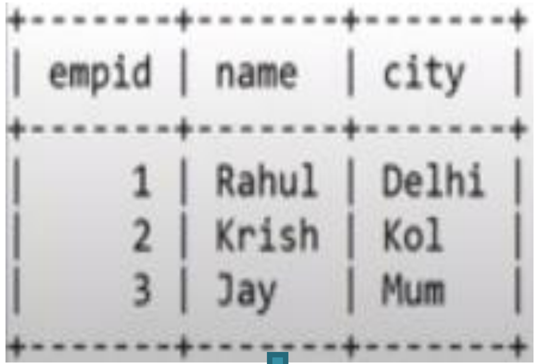
Ex: -

```
SELECT name FROM Student
UNION
SELECT name FROM Employee;
```

Aggregate means min, max, count etc function

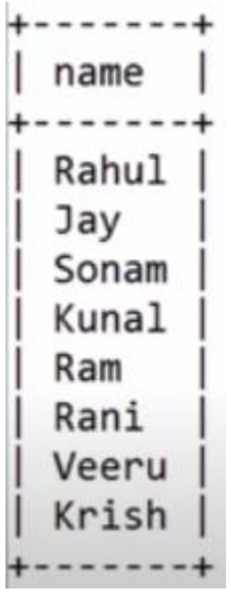


```
+-------+-------+-------+
| stuid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Jay   | Kol   |
|     3 | Sonam | Delhi |
|     4 | Kunal | Hyd   |
|     5 | Ram   | Delhi |
|     6 | Rani  | Patna |
|     7 | Veeru | Kol   |
+-------+-------+-------+
```

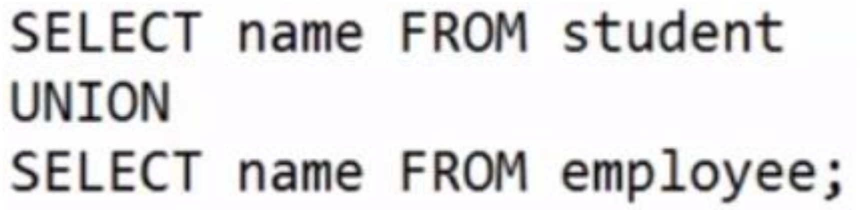


```
+-------+-------+-------+
| empid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Krish | Kol   |
|     3 | Jay   | Mum   |
+-------+-------+-------+
```

OR

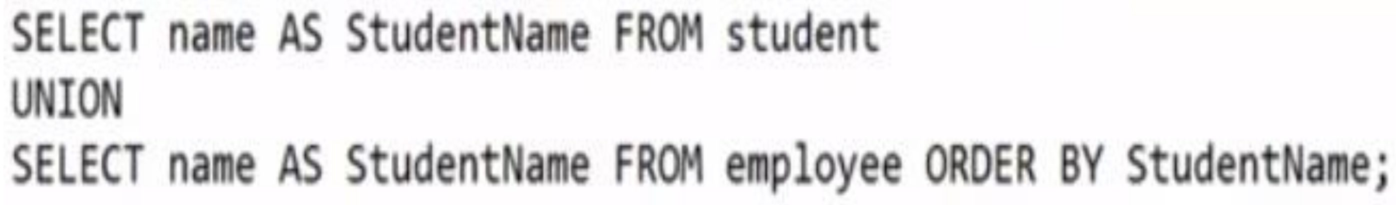


```
+-------+
| name  |
+-------+
| Rahul |
| Jay   |
| Sonam |
| Kunal |
| Ram   |
| Rani  |
| Veeru |
| Krish |
+-------+
```

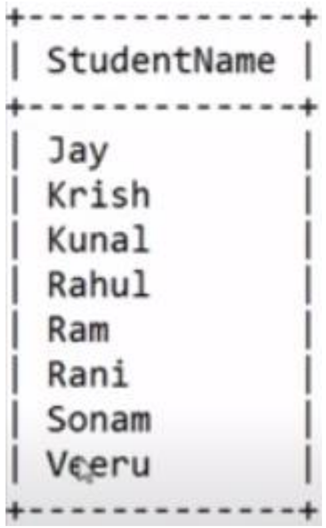


```
SELECT name FROM student
UNION
SELECT name FROM employee;
```

Sorting StudentName in Ascending Order



```
SELECT name AS StudentName FROM student
UNION
SELECT name AS StudentName FROM employee ORDER BY StudentName;
```



```
+-------------+
| StudentName |
+-------------+
| Jay         |
| Krish       |
| Kunal       |
| Rahul       |
| Ram         |
| Rani        |
| Sonam       |
| Veeru       |
+-------------+
```

Set Difference

- The set difference of two relations R and S (R – S) is the set of all tuples that are in R but not in S.

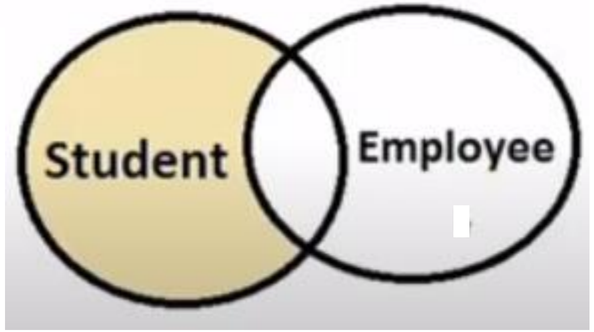
- In this case, not only should R and S be union compatible, but the operation is also asymmetric (i.e., R – S ≠S – R).

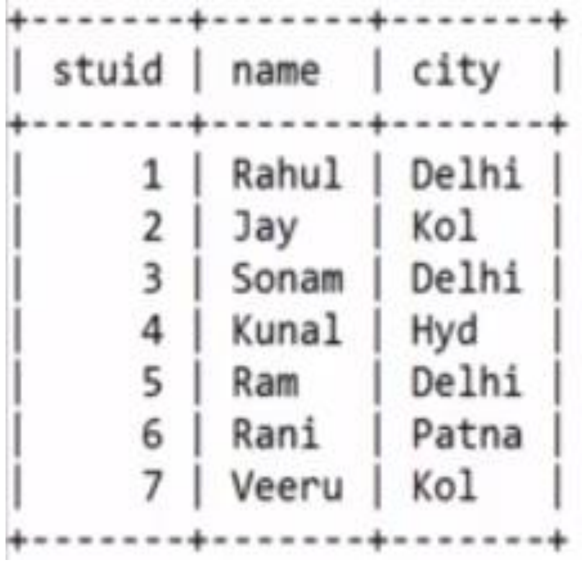
- This operation allows the deletion of tuples from a relation. Together with the union operation, we can perform modification of tuples by deletion followed by insertion.

Set Difference

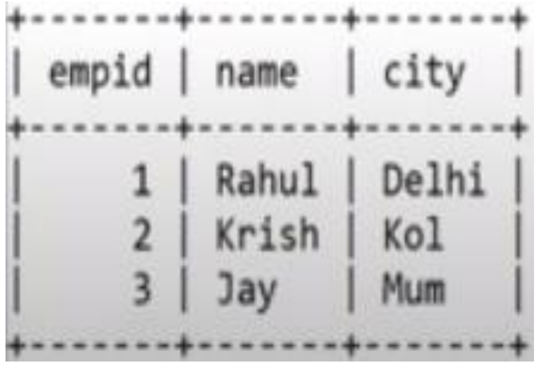
EXCEPT returns only those columns that are in the first query but not in the second query.

```sql
SELECT name FROM Student
EXCEPT
SELECT name FROM Employee;
```





```
+-------+-------+-------+
| stuid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Jay   | Kol   |
|     3 | Sonam | Delhi |
|     4 | Kunal | Hyd   |
|     5 | Ram   | Delhi |
|     6 | Rani  | Patna |
|     7 | Veeru | Kol   |
+-------+-------+-------+
```



```
+-------+-------+-------+
| empid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Krish | Kol   |
|     3 | Jay   | Mum   |
+-------+-------+-------+
```

EXCEPT

Name
Sonam
Kunal
Ram
Rani
Veeru

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

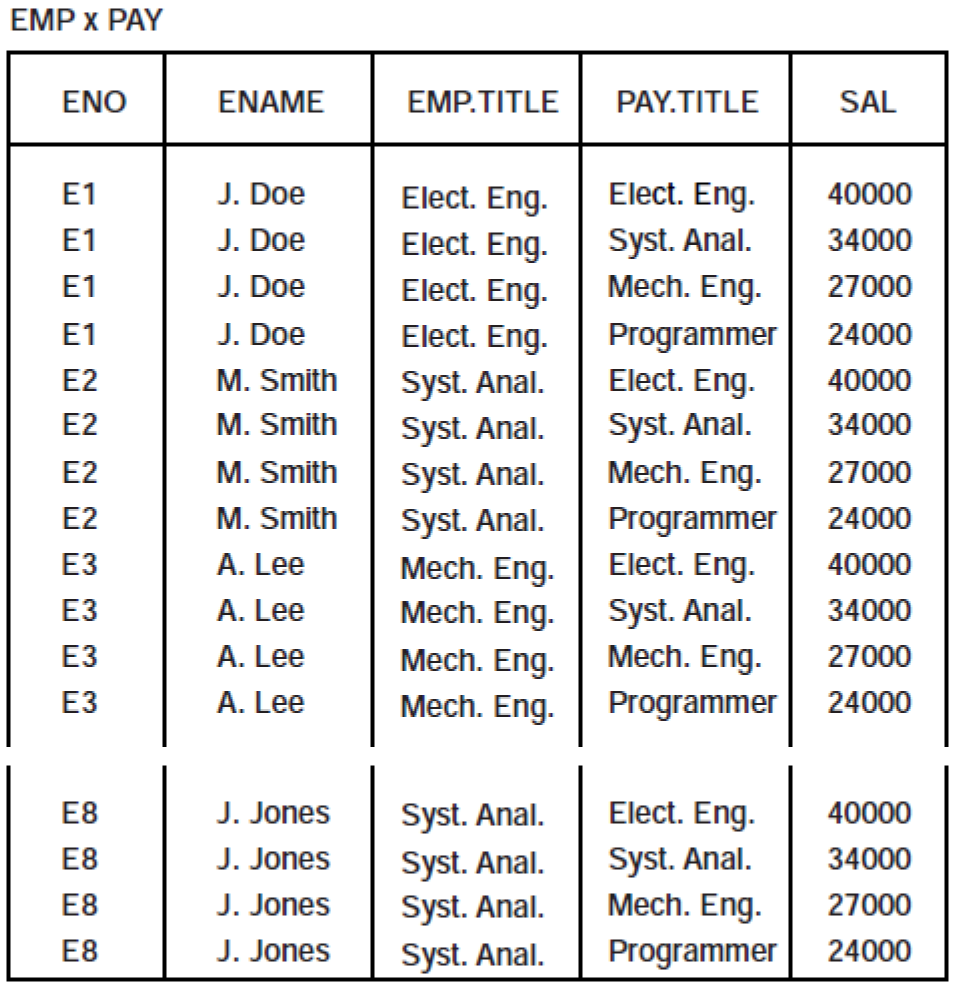
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Cartesian Product

- The Cartesian product of two relations R of degree k_1 and S of degree k_2 is the set of (k_1+k_2)-tuples, where each result tuple is a concatenation of one tuple of R with one tuple of S, for all tuples of R and S.

- The Cartesian product of R and S is denoted as R x S.

- It is possible that the two relations might have attributes with the same name. In this case the attribute names are prefixed with the relation



EMP x PAY

ENO	ENAME	EMP.TITLE	PAY.TITLE	SAL
E1	J. Doe	Elect. Eng.	Elect. Eng.	40000
E1	J. Doe	Elect. Eng.	Syst. Anal.	34000
E1	J. Doe	Elect. Eng.	Mech. Eng.	27000
E1	J. Doe	Elect. Eng.	Programmer	24000
E2	M. Smith	Syst. Anal.	Elect. Eng.	40000
E2	M. Smith	Syst. Anal.	Syst. Anal.	34000
E2	M. Smith	Syst. Anal.	Mech. Eng.	27000
E2	M. Smith	Syst. Anal.	Programmer	24000
E3	A. Lee	Mech. Eng.	Elect. Eng.	40000
E3	A. Lee	Mech. Eng.	Syst. Anal.	34000
E3	A. Lee	Mech. Eng.	Mech. Eng.	27000
E3	A. Lee	Mech. Eng.	Programmer	24000
E8	J. Jones	Syst. Anal.	Elect. Eng.	40000
E8	J. Jones	Syst. Anal.	Syst. Anal.	34000
E8	J. Jones	Syst. Anal.	Mech. Eng.	27000
E8	J. Jones	Syst. Anal.	Programmer	24000

Intersection

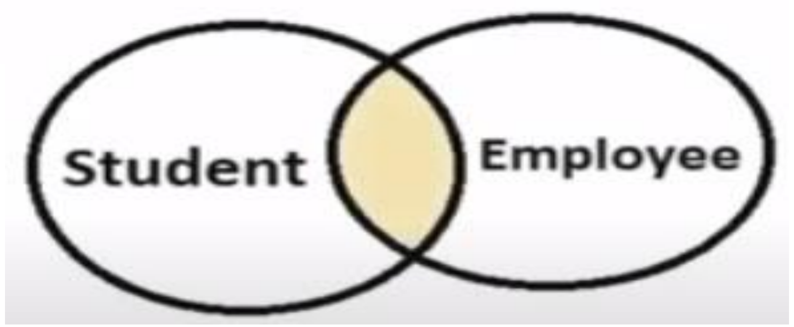
- Intersection of two relations R and S (R ∩ S) consists of the set of all tuples that are in both R and S.
- In terms of the basic operators, it can be specified as follows:

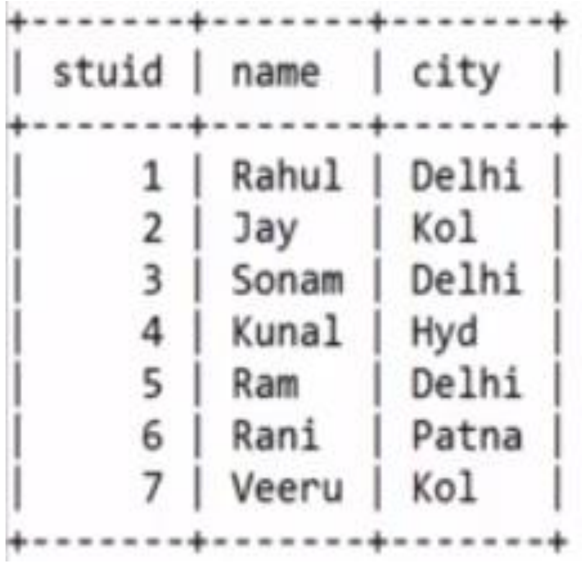
$$R \cap S = R - (R - S)$$

INTERSECTION

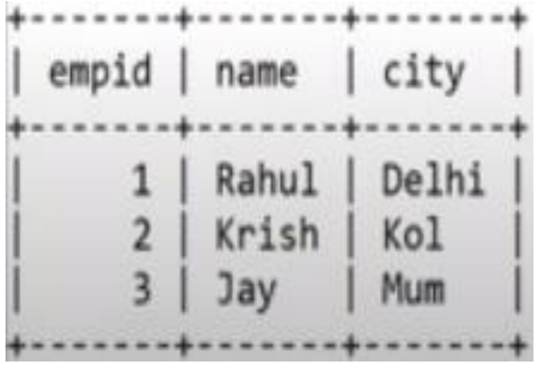
INTERSECT returns only those columns that are in the first query and also in the second query.

```
SELECT name FROM Student
INTERSECT
SELECT name FROM Employee;
```



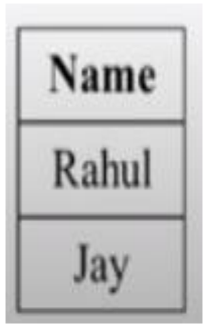


```
+-------+-------+-------+
| stuid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Jay   | Kol   |
|     3 | Sonam | Delhi |
|     4 | Kunal | Hyd   |
|     5 | Ram   | Delhi |
|     6 | Rani  | Patna |
|     7 | Veeru | Kol   |
+-------+-------+-------+
```



```
+-------+-------+-------+
| empid | name  | city  |
+-------+-------+-------+
|     1 | Rahul | Delhi |
|     2 | Krish | Kol   |
|     3 | Jay   | Mum   |
+-------+-------+-------+
```

AND



Name
Rahul
Jay

θ-Join

- Join is a derivative of Cartesian product.
- The most general type of inner join is the **θ** -join. The **θ** -join of two relations R and S is denoted as

$$R \bowtie_F S$$

- The join of two relations is equivalent to performing a selection, using the join predicate as the selection formula, over the Cartesian product of the two operand relations. Thus

$$R \bowtie_F S = \sigma_F(R \times S)$$

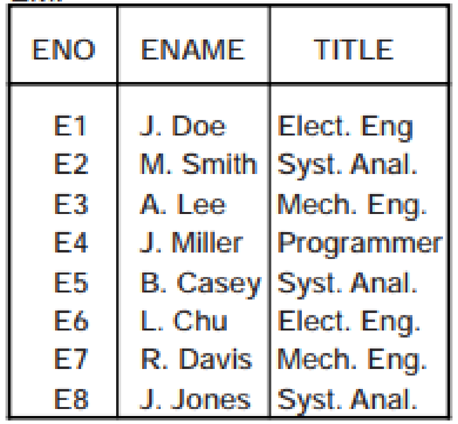
Let us consider that the EMP relation in Figure and add two more tuples Then Figure on right shows the **θ** -join of relations EMP and ASG over the join predicate

EMP.ENO = ASG.ENO.

The same result could have been obtained as

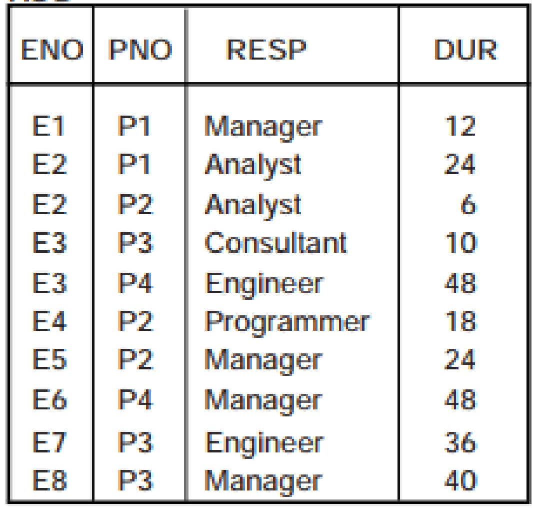
$$EMP \bowtie_{EMP.ENO=ASG.ENO} ASG =$$
$$\Pi_{ENO,\ ENAME,\ TITLE,\ SAL}(\sigma_{EMP.ENO\ =PAY.ENO}(EMP \times ASG))$$

EMP



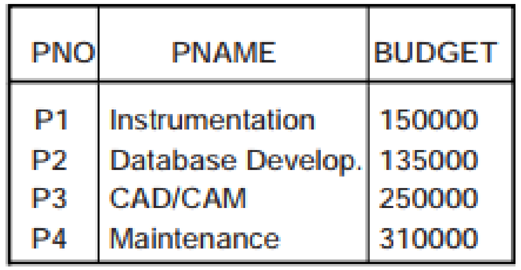
ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG



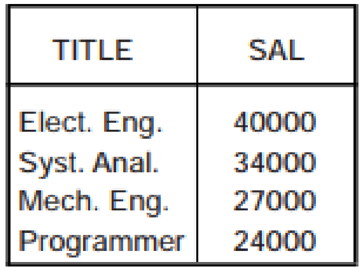
ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ



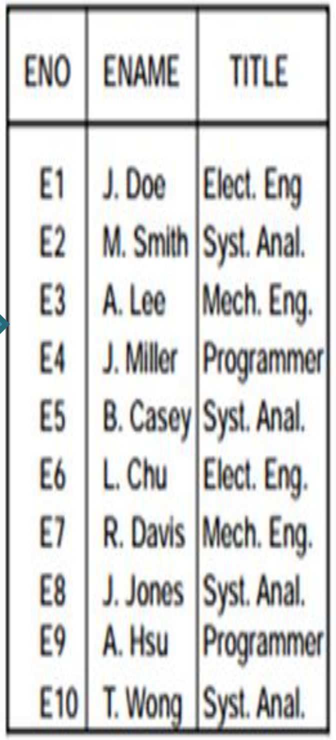
PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY



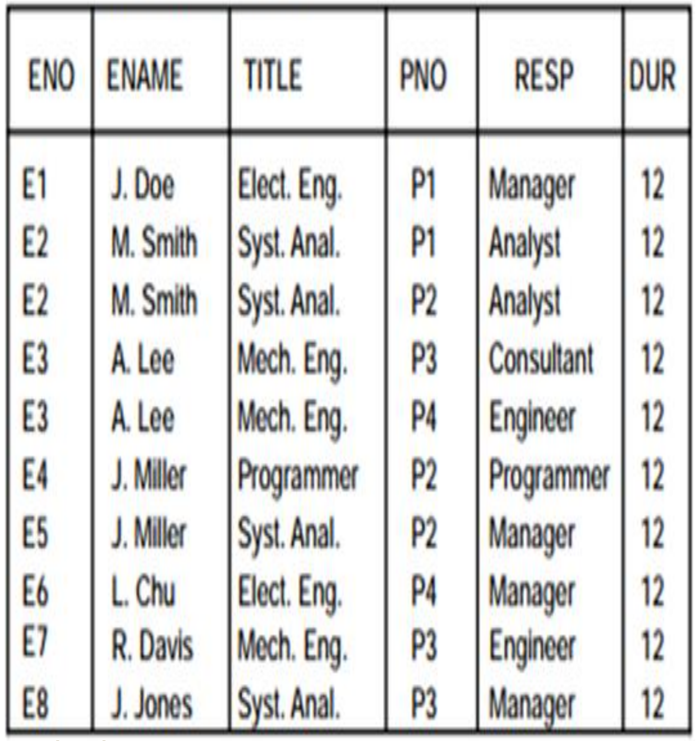
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

EMP



ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.
E9	A. Hsu	Programmer
E10	T. Wong	Syst. Anal.

$EMP \bowtie_{EMP.ENO=ASG.ENO} ASG$



ENO	ENAME	TITLE	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	P1	Manager	12
E2	M. Smith	Syst. Anal.	P1	Analyst	12
E2	M. Smith	Syst. Anal.	P2	Analyst	12
E3	A. Lee	Mech. Eng.	P3	Consultant	12
E3	A. Lee	Mech. Eng.	P4	Engineer	12
E4	J. Miller	Programmer	P2	Programmer	12
E5	J. Miller	Syst. Anal.	P2	Manager	12
E6	L. Chu	Elect. Eng.	P4	Manager	12
E7	R. Davis	Mech. Eng.	P3	Engineer	12
E8	J. Jones	Syst. Anal.	P3	Manager	12

Join Types

- Inner join requires the joined tuples from to satisfy the join predicate.
- In contrast, outer join does not have this requirement – tuples exist in the result relation regardless.
- Outer join can be of three types: Left outer join ,Right outer join, Full outer join.
- In the left outer join, the tuples from the left operand relation are always in the result, in the case of right outer join, the tuples from the right operand are always in the result, and in the case of full outer join, tuples from both relations are always in the result.
- Outer join is useful in those cases where we wish to include information from one or both relations even if the do not satisfy the join predicate.

Natural Join

- A natural join is an equi-join of two relations over a specified attribute, more specifically, over attributes with the same domain.
- There is a difference, however, in that usually the attributes over which the natural join is performed appear only once in the result.
- A natural join is denoted as the join without the formula.

$$R \bowtie_A S$$

Figure 2.8 shows the natural join of relations EMP and PAY in Figure 2.3 over the attribute TITLE

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

EMP $\bowtie_{\text{TITLE}}$ PAY

ENO	ENAME	TITLE	SAL
E1	J. Doe	Elect. Eng.	40000
E2	M. Smith	Analyst	34000
E3	A. Lee	Mech. Eng.	27000
E4	J. Miller	Programmer	24000
E5	B. Casey	Syst. Anal.	34000
E6	L. Chu	Elect. Eng.	40000
E7	R. Davis	Mech. Eng.	27000
E8	J. Jones	Syst. Anal.	34000

Left Outer Join

- Consider the left outer join of EMP and ASG over attribute ENO

$$EMP \quad {}_{ENO}\, ASG)$$

- Notice that the information about two employees, E9 and E10 are included in the result even thought they have not yet been assigned to a project with "Null" values for the attributes from the ASG relation.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

$EMP ⟕_{ENO} ASG$

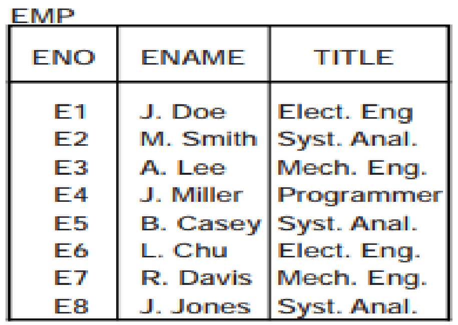
ENO	ENAME	TITLE	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	P1	Manager	12
E2	M. Smith	Syst. Anal.	P1	Analyst	12
E2	M. Smith	Syst. Anal.	P2	Analyst	12
E3	A. Lee	Mech. Eng.	P3	Consultant	12
E3	A. Lee	Mech. Eng.	P4	Engineer	12
E4	J. Miller	Programmer	P2	Programmer	12
E5	J. Miller	Syst. Anal.	P2	Manager	12
E6	L. Chu	Elect. Eng.	P4	Manager	12
E7	R. Davis	Mech. Eng.	P3	Engineer	12
E8	J. Jones	Syst. Anal.	P3	Manager	12
E9	A. Hsu	Programmer	Null	Null	Null
E10	T. Wong	Syst. Anal.	Null	Null	Null

Semijoin

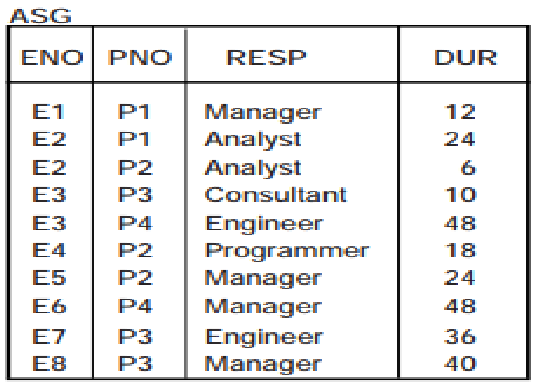
- The semijoin of relation *R*, defined over the set of attributes *A*, by relation *S*, defined over the set of attributes *B*, is the subset of the tuples of *R* that participate in the join of *R* with *S*.

$$R \ltimes_F S = \Pi_A(R \bowtie_F S) = \Pi_A(R) \bowtie_F \Pi_{A\cap B}(S)$$
$$= R \bowtie_F \Pi_{A\cap B}(S)$$

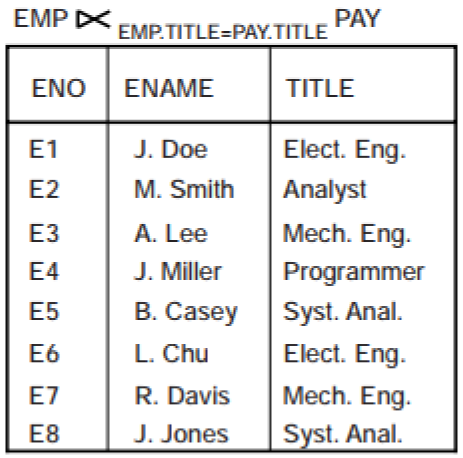
- The advantage of semijoin is that it decreases the number of tuples that need to be handled to form the join.
- In centralized database systems, this is important because it usually results in a decreased number of secondary storage accesses by making better use of the memory.
- It is even more important in distributed databases since it usually reduces the amount of data that needs to be transmitted between sites in order to evaluate a query.


EMP

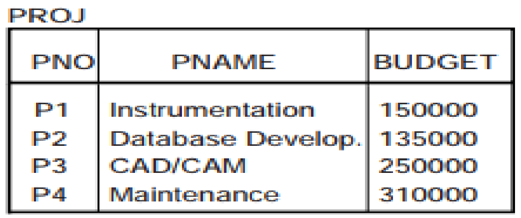
ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.


ASG

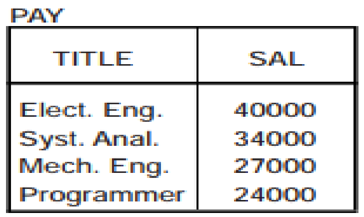
ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40


$\text{EMP} \ltimes_{\text{EMP.TITLE=PAY.TITLE}} \text{PAY}$

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Analyst
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.


PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000


PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Relational Calculus

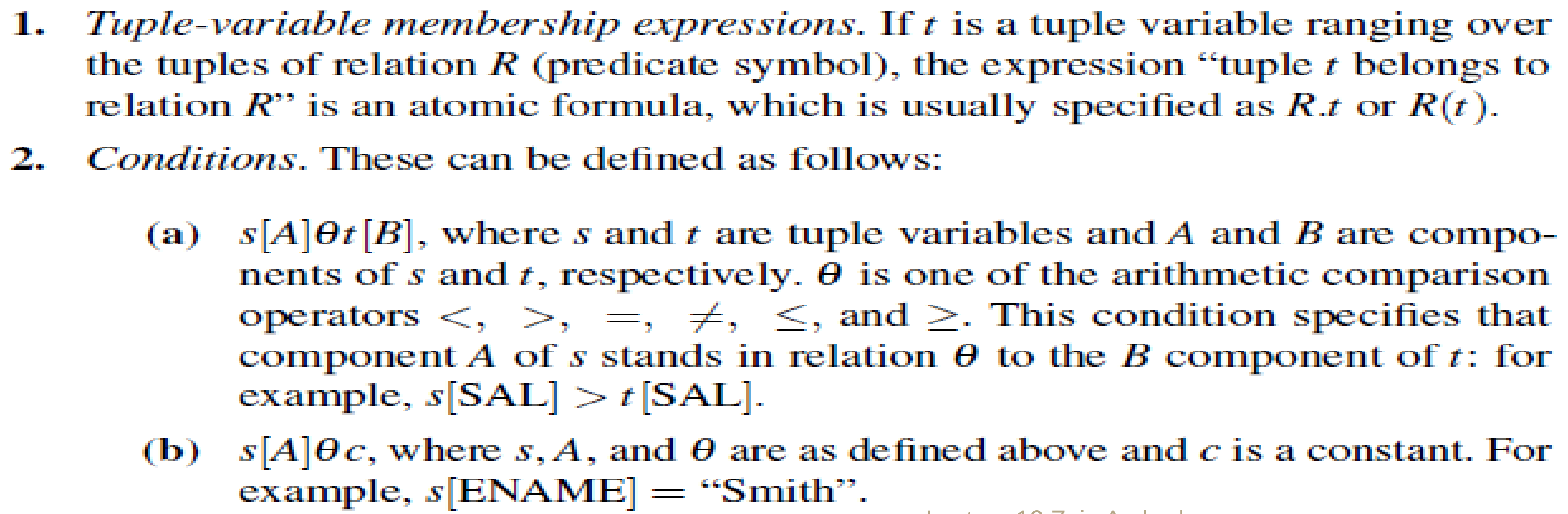
- In relational calculus-based languages, instead of specifying how to obtain the result, one specifies what the result is by stating the relationship that is supposed to hold for the result.

Relational calculus languages fall into two groups:

1. **Tuple relational**
2. **Calculus and domain relational calculus.**

Tuple relational calculus

- The primitive variable used in tuple relational calculus is a tuple variable which specifies a tuple of a relation.
- In tuple relational calculus queries are specified as $\{t | F(t)\}$
- where t is a tuple variable and F is a well-formed formula.



1. *Tuple-variable membership expressions.* If t is a tuple variable ranging over the tuples of relation R (predicate symbol), the expression "tuple t belongs to relation R" is an atomic formula, which is usually specified as $R.t$ or $R(t)$.
2. *Conditions.* These can be defined as follows:

   (a) $s[A]\theta t[B]$, where s and t are tuple variables and A and B are components of s and t, respectively. θ is one of the arithmetic comparison operators $<$, $>$, $=$, $\neq$, $\leq$, and $\geq$. This condition specifies that component A of s stands in relation θ to the B component of t: for example, $s[\text{SAL}] > t[\text{SAL}]$.

   (b) $s[A]\theta c$, where s, A, and θ are as defined above and c is a constant. For example, $s[\text{ENAME}] =$ "Smith".

Domain relational calculus

- The fundamental difference between a tuple relational language and a domain relational language is use of a domain variable in the latter.
- A domain variable ranges over the values in a domain and specifies a component of a tuple. In other words, the range of a domain variable consists of the domains over which the relation is defined.
- The queries are specified in the following form

$$x_1, x_2, \ldots, x_n \mid F(x_1, x_2, \ldots, x_n)$$

where F is a wff in which $x_1, \ldots, x_n$ are the free variables